Applied and
Computational
Mathematics
Division

NISTIR 4475

Center for Computing and Applied Mathematics

The Guide to Available Mathematical Software Problem Classification System

*Ronald F. Boisvert, Sally E. Howe
and David K. Kahaner*

November 1990

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899

NISTIR 4475

THE GUIDE TO AVAILABLE MATHEMATICAL SOFTWARE PROBLEM CLASSIFICATION SYSTEM

Ronald F. Boisvert
Sally E. Howe
David K. Kahaner

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Center for Computing and Applied Mathematics
Gaithersburg, MD 20899

November 1990

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

own manuals or on-line documentation system. In order to determine what software is available to solve a particular problem, users must search through a very large, heterogeneous collection of information. This is a tedious and error-prone process. As a result, there has been much interest in the development of automated advisory systems to help users select software.

Keyword search is a popular technique used for this purpose. In such a system keywords or phrases are assigned to each piece of software to succinctly define its purpose, and the set of all such keywords are entered into a database. Keyword-based selection systems query users for a set of keywords and then present a list of software modules which contain them. A major difficulty with such systems is that users often have trouble in providing the appropriate keywords for a given mathematical or statistical problem. There is such a wealth of alternate mathematical and statistical terminology that it would be a rare occurrence for two separate knowledgeable persons to assign the same set of keywords to a given software module. Users of these systems, who are usually much less familiar with the terminology, often find that there are either too many software modules associated with the keywords that they have specified, or none. One can attempt to ameliorate this problem by imposing a standard set of keywords or by implementing a very elaborate keyword-specification scheme for users. The former is difficult to maintain and the latter may not be easy to use.

Classification systems have long been used to give structure to large bodies of information. A well-formulated system can improve understanding of the information as well as ease access to it, thus making the information more useful. The Dewey Decimal System, for example, provides a means for librarians to maintain a large collection of books. Since the system is subject-oriented, library users can quickly find books in a given subject area. Likewise, a subject-oriented classification system can be an effective means of directing users to appropriate mathematical and statistical software.

To be effective, such a classification system must have the following properties.

1. **Problem-orientation**. It must classify the *problems* which can be solved by computer software. Other orientations, such as classification by algorithm or classification by software package, are of less interest to end users.

2. **Variable-level tree structure**. A tree structure is the most natural for a classification system. Allowing arbitrary levels of refinement permits the system to adapt to both mature and young subject areas. In young subject areas little software is available, and hence little refinement is necessary. In mature areas where much software is available, increased refinement is necessary to distinguish among the choices.

3. **Active maintenance**. The system must be monitored and revised over time to best reflect the current state of the rapidly growing mathematical and statistical software collection. The tree structure facilitates this by insuring that modifications of the system are localized.

The classification system can be thought of as a pre-defined hierarchy of keywords. Since the entire universe of these keywords is visible to both developers and users the possibility

of finding the correct software is improved. In addition, the tree structure shows explicit relationships among the set of keywords which can aid in users' understanding of them.

To use such a system, each piece of software must be assigned a class. Classification-based software selection systems permit users to incrementally refine their specification of the problem by using the classification system as a decision tree. When the most appropriate problem class has been selected by the user, software which contains the selected classification is presented.

Classification systems are not themselves free of problems. For example, they partition software much more coarsely than do keywords, and users may not be fluent in the terminology of the classification system. Nevertheless, we believe that the structure imposed by such systems improves both the user's access to information and the implementor's job of maintaining it.

In this paper we describe a particular classification system for mathematical and statistical software which meets the criteria described above. The system is an outgrowth of the Guide to Available Mathematical Software (GAMS) project at NIST. We begin by describing the origins of system, and then outline its current version. We next discuss how such a system can be effectively used for software selection. The entire classification system is included in Appendix A. Appendix B is devoted to summarizing the differences between the current version and its last widely-publicized version.

2 Origins of the System

The system described in this paper has its origins in a software classification scheme devised in the 1960s by SHARE, the IBM Users Group. The SHARE scheme had a fixed two-level structure which led to very broad classes. Such a coarse structure was not suited to classifying very large software collections.

In 1975 John Bolstad proposed a substantial revision of the SHARE scheme which eliminated many of its weaknesses [7]. The Bolstad system was a multi-level tree-structured scheme. Unfortunately, it attempted to maintain compatibility with the SHARE scheme wherever possible and hence it inherited much of the SHARE system's illogical organization. Nevertheless, the scheme was a great improvement and versions of it were adopted for use by a number of groups, including NBS[1].

Many difficulties in using the Bolstad scheme surfaced at NBS while attempting to classify about 2500 subprograms, representing most of the widely-distributed mathematical software then available. As a result, many new classifications were added and some sections were completely reworked. The resulting scheme was used in the first GAMS software catalog [5] and was adopted for use in the documentation of the SLATEC[2] Common Math Library [10].

In 1983 Boisvert, Howe and Kahaner completely revised the classification system used in GAMS and published it for public review [2]. This became known as version 1.0 of the GAMS Classification Scheme for mathematical and statistical software. The new system kept

[1]National Bureau of Standards. NBS became NIST in 1988.

[2]Sandia–Los Alamos–Air Force Weapons Laboratory Technical Exchange Committee

the Bolstad philosophy while providing an organization which more accurately reflected the then current state of mathematical and statistical software. In addition, many sections of the Bolstad scheme not directly related to mathematical or statistical software were deleted. Instead, the new scheme was viewed as a node in a larger scheme which encompassed all computer software.

Minor modifications of the GAMS scheme have appeared in [3] (version 1.1), [4] (version 1.2), and [8] (version 1.3). Since 1983, the scheme has been adopted for use by a number of institutions, including Amoco Production Research, ASA Statistical Computing Section Committee on Statistical Algorithms, C. Abaci (The Scientific Desk), Centro di Calccolo Elettronico Interuniversitario dell'Italia Nord-orientale (CINECA, Bologna), Eigenössische Technische Hochschule Zürich (ETH) Seminar für Angewandte Mathematik, IMSL Inc., Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Los Alamos National Laboratories, National Center for Atmospheric Research, SLATEC Common Math Library Subcommittee, Stanford Linear Accelerator Center (SLAC), State University of Utrecht Academic Computer Center, and University of Texas System Center for High Performance Computing.

3 Version 2 of the Classification System

Much new mathematical and statistical software has appeared since the GAMS classification system was developed. In many cases this software addresses problems not explicitly included in the original classification system. In other cases, significant new software packages have appeared which have provided improved methods of organizing certain subject areas. Because of this we have again found it necessary to modify the GAMS Classification System. The resulting scheme, termed version 2.0, appeared in the recently published GAMS software catalog [6], and is reproduced in its entirety in Appendix A of this paper. Appendix B outlines the differences between this version and Version 1.2.

The highest levels of the classification system have remain unchanged since version 1.0. In the following we describe the purpose of each.

- *A. Arithmetic, error analysis*
  Contains software implementing elementary arithmetic operations on non-standard data types. Examples are extended precision arithmetic and interval arithmetic. Also included are systems or utilities which do general-purpose error analysis. Finally, software for accelerating the convergence of sequences is also found here.

- *B. Number theory*
  Software classified here performs such number-theoretic calculations as the decomposition of integers into prime factors.

- *C. Elementary and special functions*
  Software for evaluating both elementary and specialized mathematical functions is found here. Examples of elementary functions are trigonometric functions, exponentials, and polynomials. Examples of special functions are Bessel functions and Gamma functions. Statistical functions such as probability density functions are found in class L5.

- *D. Linear Algebra*
  This class includes elementary vector and matrix operations, matrix factorizations, solution of linear systems, eigenvalue problems, determinants, and inverses.

- *E. Interpolation*
  Software for finding a function which "passes through" given data values in one or more dimensions is found here. If the data have noise then classes K or L8 are more appropriate.

- *F. Solution of nonlinear equations*
  This class contains software for solving systems of nonlinear equations. Software for single nonlinear equations and polynomial equations are also included.

- *G. Optimization*
  Software for minimizing or maximizing functions with or without constraints is found here. This includes linear programming, nonlinear programming, integer programming, network optimization, and optimal control.

- *H. Differentiation, integration*
  Here one finds software for estimating derivatives and evaluating integrals.

- *I. Differential and integral equations*
  Software for solving ordinary differential equations, partial differential equations, and integral equations is found here.

- *J. Integral transforms*
  This class includes software for Fourier transforms, trigonometric transforms, Laplace transforms, Hilbert transforms, convolutions, etc.

- *K. Approximation*
  Software for determining best approximations to functions or data in various norms (e.g., L_1, L_2, L_∞) are classified here. Software for approximation followed by statistical analysis (e.g., regression) is classified in L8. Software for solving linear algebraic systems (e.g. solution of overdetermined systems in the least squares sense) is classified in D9. Software for interpolation is classified in E.

- *L. Statistics, probability*
  Software for statistical computing is classified here. This includes data summarization and manipulation, elementary data analysis (e.g., calculating the sample mean), statistical graphics, statistical function evaluation, random number generation, analysis of variance, regression, categorical data analysis, time series analysis, correlation analysis, discriminant analysis, covariance structure models, cluster analysis, and survival analysis.

- *M. Simulation, stochastic modeling*
  Software for building and studying stochastic models is classified here.

- *N. Data handling*
  Data handling includes various operations such as input, output, sorting, searching, merging, and permuting. Software implementing useful data structures such as heaps and trees is also found here.

- *O. Symbolic computation*
  Software for manipulating mathematical expressions in their symbolic form is classified here.

- *P. Computational geometry*
  This class includes software for fundamental geometric calculations (e.g., areas and volumes) and implementation of algorithms for geometric problems, such as computation of the convex hull and the Voronoi diagram.

- *Q. Graphics*
  General-purpose computer graphics is classified here. Statistical graphics is in L3.

- *R. Service routines*
  This class includes software which performs low-level utility functions such as error checking, error handling, and retrieval of information about machine characteristics.

- *S. Software development tools*
  Tools which facilitate mathematical software development and maintenance and classified here. Tool types include program transformation (e.g., convert to double precision), static analysis (e.g., flow analysis, interface analysis), and dynamic analysis (e.g., tracing, timing, assertion checking).

- *Z. Other*
  This class contains software which does not fit anywhere else.

Strictly speaking, classes N, Q, R, and S are not mathematics or statistics. They have been included because such software is commonly found in mathematical and statistical packages.

4 Using the System

Version 2.0 of the GAMS Classification System has been used to classify more than 5000 software modules from 40 separate packages at NIST. The 1990 edition of the *Guide to Available Mathematical Software* [6] lists all of these modules in order of their classifications. A software advisory system called the GAMS Interactive Consultant (GAMSIC) provides on-line access to this same information [1]. Such successful application indicates that the system adequately reflects the current state of mathematical and statistical software development. Some of the well-known packages which have been classified are listed in Table 1. In what follows we discuss some of the guidelines which have governed the use of the system at NIST.

Table 1: Some well-known packages classified at NIST

Package	n†	Description
BMDP	43	Programs for statistical data analysis.
CALGO	172	The Collected ALGOrithms of the ACM. Programs published by the ACM Transactions on Mathematical Software. (1975-88)
CMLIB	763	The NIST Core Math Library. A collection of public-domain Fortran subroutine packages. (Includes LINPACK, EISPACK, FISHPAK, QUADPACK, FFTPKG, etc., many of which are also found in the SLATEC library)
DATAPAC	169	Fortran subprograms for statistical data analysis.
Dataplot	87	Interactive graphical and statistical data analysis program.
IMSL	470	Fortran subprograms for mathematics and statistics. (Version 9.2)
MATH/LIBRARY	668	Fortran subprograms for mathematics from IMSL Inc. (Version 1.0)
STAT/LIBRARY	620	Fortran subprograms for statistics from IMSL Inc. (Version 1.0)
SFUN/LIBRARY	297	Fortran subprograms for evaluating special functions from IMSL Inc. (Version 2.0)
MAGEV	80	The MAth/GEophysical Vector library. Fortran subprograms developed for the Cyber 205. (Version 3.3)
MINITAB	63	Program for statistical data analysis.
NAG	774	Fortran subprograms for mathematics and statistics. (Mark 13)
NMS	50	Fortran subprograms from [9].
PORT	327	Fortran subprograms for mathematics. (Version 2)
SAS	40	Program for statistical data analyses. (Version 15.8)
Scientific Desk	329	Fortran subprograms for mathematics and statistics for use on PC's. (Version 4)
SPSS	28	Program for statistical data analysis. (Version 2.2)
STARPAC	145	Fortran subprograms for statistical data analysis. (Version 2.07)

† n is the number of modules classified.

4.1 Guidelines for Classifying Modules

When using the classification system one must decide what objects to classify and then how to assign classifications. The following guidelines were used to classify software modules in the GAMS catalog.

Modules may represent different types of objects.

The use of the GAMS Classification System is most straightforward in the case of subprogram libraries, where one classifies the individual user-callable subprograms. However, most statistical software, and an increasing amount of mathematical software, comes in the form of stand-alone programs with their own input command languages. When such a program is designed to solve a very restricted set of problems, then it is reasonable to classify it just as a subprogram would be. This is the case, for example, with the programs in the BMDP package. In other cases, a single program may be capable of solving a very wide range of problems; interactive statistical analysis systems like Dataplot and Minitab are examples. In such cases one is faced with a dilemma: one must either classify the program at a very high level (where it will likely not be found) or give it many classifications (each of which provides little information). We have chosen instead to classify the major *commands* in the input language of these programs. In this way, classification-based software advisory systems have information available about the commands available in these multi-purpose programs in the same way as user-callable subprograms for the same problem. This avoids classification at too high a level and provides catalog readers with more than the name of the program. In other cases we have chosen to classify an entire subprogram library as a single unit. This occurs in the case of Fortran-callable graphics libraries. We wish to catalog the available graphics libraries in GAMS, but the classification system is not yet sufficiently refined to meaningfully classify the individual subprograms. In addition, since users rarely use more than one graphics library at a time, further refinement is less critical than with mathematical and statistical libraries.

Some modules should not be classified.

It is a mistake to attempt to classify every software module in a given package, even when each module is "user-callable". Some modules, although called by the user, are subsidiary in nature. Examples of these are modules which are called to change defaults or perform some initialization in preparation for the use of another module. Other examples are modules which evaluate fitted functions or interpolate the computed solution to a differential equation. The detailed documentation of the module which solves the main problem of interest will point to such subsidiary routines. Classifying these would only add needless clutter to a software catalog.

Modules may be classified at more than one node.

Many software modules have multiple purposes, and hence should be assigned multiple classes. Also, since there is considerable overlap in many areas of the classification scheme,

it may be difficult to assign a unique classification in every case. For example, nonlinear least squares approximation (K1b) and nonlinear regression (L8b) are the same problem seen from different points of view.

It is interesting to note that the assignment of alternate classifications to modules is context-sensitive, i.e., it depends on the collection of software being classified. For example, consider the relationship between software for nonlinear least squares problems and software for nonlinear regression. Both solve the same basic mathematical problem, but the latter software takes a statistical point of view (i.e., it uses the terminology of statistics and returns additional statistical information which can be used to judge goodness of fit). If the software collection is rich in codes of both types, then nonlinear regression codes should only be classified in subtree L8. Users with a statistical orientation will natural go down the L8 path to find such codes; assigning L8 classes to nonlinear least squares codes will only complicate the selection process. On the other hand, if the software collection contains few codes for nonlinear regression, then assigning classes in L8 to the nonlinear least squares codes might be the only way that naive users could discover software appropriate for nonlinear regression.

Modules may be classified at any node in the tree.

We classify modules at the lowest level of the tree which accurately describes the problem solved. This gives the best match between classification and software. In some cases the node selected in this manner is not a leaf of the tree. This situation occurs when there is no child node which adequately describes the function of the module. Classification at non-leaf nodes can also be done when a module solves all (or at least most) of the problems given at the lower levels. This is not generally recommended however, since users will tend to look as far down into the tree as possible in locating their problems.

4.2 Associated Materials

In order to make best use of the classification system in the development of the GAMS catalog, we have developed some associated materials. Among these are a verbose realization of the classification system and a keyword index to the classification system.

The wording of classes given in Appendix A is appropriate for use when the entire system is displayed in outline form as it is there. The problem descriptions are quite terse and are meant to be read in context. In some cases it is necessary to use such descriptions out of context, however. For example, when modules are listed in order of classifications in the GAMS catalog, the parent classification is rarely found on the same page as a given classification. Similarly, GAMSIC displays the current class and its children, prompting the user for the next subclass to go to. In each of these cases the terse class description may not be enough for a reader to determine what problem the current class represents. As a result of this we maintain two separate versions of the GAMS classification system which we refer to as *terse* and *verbose*. The terse version is given in Appendix A. The verbose descriptions provide enough additional wording so that the problem represented by each class can be determined out of context. The latter can be found in the *Modules by Class* section of [6].

One of the roadblocks encountered by users in trying to find software using the GAMS Classification System is that the terminology used to describe mathematical and statistical problems may not be familiar to them. A partial solution developed for the GAMS catalog has been the development of a keyword index to the classification system. Not only does this allow us to provide pointers into the classification system using alternate terminology, but it also provides users a faster method of getting close to a desired class than a linear search on paper or an automated tree traversal. The index which we have developed can be found in [6].

5 Future Development

Since its creation nearly a decade ago, the GAMS Classification System has undergone substantial enhancement and revision. There is still much work to be done. In this section we describe some of our ideas for future editions.

Many leaves of the C subtree (Elementary and Special Functions) combine a number of related functions in a single class. For example, Bessel functions J, Y, H_1, and H_2 are all in a single class as are the Airy functions Ai and Bi. The amount of software available for the evaluation of special functions is steadily increasing, as can be seen in the current GAMS catalog. In order to reduce the number of modules in these classes to a more manageable size it may be necessary to refine many of the subtrees class C.

There is no reasonable place to classify software for the manipulation of piecewise polynomials (splines) in the current scheme. Such software is currently classified in subtrees E6 (Service routines for interpolation) or K6 (Service routines for approximation). These classes primarily were designed to contain subsidiary software associated with interpolation and approximation. The existence of these classes violates our guideline about not classifying subsidiary software modules, and hence we intend to delete these two subtrees. However, we must find a new home for general-purpose programs for the manipulation of piecewise polynomials. We believe that such a class probably belongs in C (Elementary and special functions) in parallel to class C3 (Polynomials).

The amount of software for solving problems in linear algebra is still increasing, and many new problems are being addressed by software in this area. Examples are: software for elementary vector and matrix operations not listed in D1 (e.g. scalar addition/subtraction, distance between vectors, angles between vectors, bilinear forms), and software for the solution of specialized types of linear systems (e.g. Toeplitz, block tridiagonal). Another problem is that software for computing matrix factorizations and inverses are found in the same classes as modules for solving systems of linear equations. This nearly triples the number of modules in these classes which are already among the most heavily populated in the GAMS catalog. Subtree D may need a substantial revision to alleviate these problems.

The subtree I1a (Initial-value problems for ordinary differential equations) violates our philosophy of partitioning by problem rather than by solution method. The growing collection of software for this problem area requires us to find better ways to partition it.

Finally, important classes remain unrefined, including Q (Graphics) and O (Symbolic Computation).

6 Conclusions

Further development of mathematical software advisory systems is necessary in order to ease user access to the steadily increasing collection of reusable mathematical and statistical software. Tree-structured problem-oriented software classification schemes are one way for such advisory systems to systematically associate software modules with the problems they solve. Versions of the Guide to Available Mathematical Software Classification System have been successfully used for this purpose for about ten years.

We seek constructive criticism of our system, especially from those who have used it to classify software. Numerous changes for the system are already being planned; are seeking interested parties to review them. Machine-readable copies of the system are available from the authors, as well as our classifications for the libraries listed in Table 1.

Disclaimer

Certain commercial products are identified in this report in order to adequately document the development and evaluation of the GAMS classification system. Identification of these products does not imply recommendation or endorsement by NIST, nor does it imply that the identified products are necessarily the best available for the purpose.

References

[1] R. F. Boisvert. The Guide to Available Mathematical Software advisory system. *Math. & Comp. in Simul.*, 31:453–464, 1989.

[2] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. The GAMS classification scheme for mathematical and statistical software. *SIGNUM Newsletter*, 18(1):10–18, 1983.

[3] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. Guide to Available Mathematical Software. NBSIR 84-2824, National Bureau of Standards, 1984. Also available as PB84-171305 from the National Technical Information Service (NTIS), Springfield, VA 22161.

[4] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. GAMS—a framework for the management of scientific software. *ACM Trans. Math. Softw.*, 11:313–355, 1985.

[5] R. F. Boisvert, S. E. Howe, D. K. Kahaner, J. Knapp-Cordes, and M. Knapp-Cordes. Guide to Available Mathematical Software. Center for Applied Mathematics, National Bureau of Standards, Washington, DC 20234, October 1981.

[6] R. F. Boisvert, S. E. Howe, D. K. Kahaner, and J. L. Springmann. Guide to Available Mathematical Software. NISTIR 90-4237, National Institute of Standards and Technology, 1990. Also available as PB90-216508/AS, National Technical Information Service, Springfield, VA 22161.

[7] J. Bolstad. A proposed classification scheme for computer program libraries. *SIGNUM Newsletter*, 10(2–3):32–39, 1975.

[8] IMSL Inc., 2500 CityWest Blvd., Houston, TX 77042-3020. *MATH/LIBRARY: FORTRAN Subroutines for Mathematical Applications*, April 1987. Version 1.0.

[9] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[10] W. Vandevender and K. Haskell. The SLATEC mathematical subroutine library. *SIGNUM Newsletter*, 17(3):16–20, 1982.

Appendix A
GAMS Classification Scheme, Version 2.0

A. Arithmetic, error analysis

A1. Integer
A2. Rational
A3. Real
A3a. Standard precision
A3c. Extended precision
A3d. Extended range
A4. Complex
A4a. Standard precision
A4c. Extended precision
A4d. Extended range
A5. Interval
A6. Change of representation
A6a. Type conversion
A6b. Base conversion
A6c. Decomposition, construction
A7. Sequences (e.g., convergence acceleration)

B. Number theory

C. Elementary and special functions *(search also class L5)*

C1. Integer-valued functions (e.g., factorial, binomial coefficient, permutations, combinations, floor, ceiling)
C2. Powers, roots, reciprocals
C3. Polynomials
C3a. Orthogonal
C3a1. Trigonometric
C3a2. Chebyshev, Legendre
C3a3. Laguerre
C3a4. Hermite
C3b. Non-orthogonal
C4. Elementary transcendental functions
C4a. Trigonometric, inverse trigonometric
C4b. Exponential, logarithmic
C4c. Hyperbolic, inverse hyperbolic
C4d. Integrals of elementary transcendental functions
C5. Exponential and logarithmic integrals
C6. Cosine and sine integrals
C7. Gamma
C7a. Gamma, log gamma, reciprocal gamma
C7b. Beta, log beta
C7c. Psi function
C7d. Polygamma function
C7e. Incomplete gamma
C7f. Incomplete beta

C7g. Riemann zeta
C8. Error functions
C8a. Error functions, their inverses, integrals, including the normal distribution function
C8b. Fresnel integrals
C8c. Dawson's integral
C9. Legendre functions
C10. Bessel functions
C10a. J, Y, H_1, H_2
C10a1. Real argument, integer order
C10a2. Complex argument, integer order
C10a3. Real argument, real order
C10a4. Complex argument, real order
C10a5. Complex argument, complex order
C10b. I, K
C10b1. Real argument, integer order
C10b2. Complex argument, integer order
C10b3. Real argument, real order
C10b4. Complex argument, real order
C10b5. Complex argument, complex order
C10c. Kelvin functions
C10d. Airy and Scorer functions
C10e. Struve, Anger, and Weber functions
C10f. Integrals of Bessel functions
C11. Confluent hypergeometric functions
C12. Coulomb wave functions
C13. Jacobian elliptic functions, theta functions
C14. Elliptic integrals
C15. Weierstrass elliptic functions
C16. Parabolic cylinder functions
C17. Mathieu functions
C18. Spheroidal wave functions
C19. Other special functions

D. Linear Algebra

D1. Elementary vector and matrix operations
D1a. Elementary vector operations
D1a1. Set to constant
D1a2. Minimum and maximum components
D1a3. Norm
D1a3a. L_1 (sum of magnitudes)
D1a3b. L_2 (Euclidean norm)
D1a3c. L_∞ (maximum magnitude)
D1a4. Dot product (inner product)
D1a5. Copy or exchange (swap)
D1a6. Multiplication by scalar
D1a7. Triad ($\alpha x + y$ for vectors x, y and scalar α)
D1a8. Elementary rotation (Givens transformation)
D1a9. Elementary reflection (Householder transformation)
D1a10. Convolutions
D1a11. Other vector operations
D1b. Elementary matrix operations
D1b1. Initialize (e.g., to zero or identity)

D1b2.	Norm
D1b3.	Transpose
D1b4.	Multiplication by vector
D1b5.	Addition, subtraction
D1b6.	Multiplication
D1b7.	Matrix polynomial
D1b8.	Copy
D1b9.	Storage mode conversion
D1b10.	Elementary rotation (Givens transformation)
D1b11.	Elementary reflection (Householder transformation)
D2.	Solution of systems of linear equations (including inversion, LU and related decompositions)
D2a.	Real nonsymmetric matrices
D2a1.	General
D2a2.	Banded
D2a2a.	Tridiagonal
D2a3.	Triangular
D2a4.	Sparse
D2b.	Real symmetric matrices
D2b1.	General
D2b1a.	Indefinite
D2b1b.	Positive definite
D2b2.	Positive definite banded
D2b2a.	Tridiagonal
D2b4.	Sparse
D2c.	Complex non-Hermitian matrices
D2c1.	General
D2c2.	Banded
D2c2a.	Tridiagonal
D2c3.	Triangular
D2c4.	Sparse
D2d.	Complex Hermitian matrices
D2d1.	General
D2d1a.	Indefinite
D2d1b.	Positive definite
D2d2.	Positive definite banded
D2d2a.	Tridiagonal
D2d4.	Sparse
D2e.	Associated operations (e.g., matrix reorderings)
D3.	Determinants
D3a.	Real nonsymmetric matrices
D3a1.	General
D3a2.	Banded
D3a2a.	Tridiagonal
D3a3.	Triangular
D3a4.	Sparse
D3b.	Real symmetric matrices
D3b1.	General
D3b1a.	Indefinite
D3b1b.	Positive definite
D3b2.	Positive definite banded
D3b2a.	Tridiagonal
D3b4.	Sparse

D3c.	Complex non-Hermitian matrices
D3c1.	General
D3c2.	Banded
D3c2a.	Tridiagonal
D3c3.	Triangular
D3c4.	Sparse
D3d.	Complex Hermitian matrices
D3d1.	General
D3d1a.	Indefinite
D3d1b.	Positive definite
D3d2.	Positive definite banded
D3d2a.	Tridiagonal
D3d4.	Sparse
D4.	Eigenvalues, eigenvectors
D4a.	Ordinary eigenvalue problems ($Ax = \lambda x$)
D4a1.	Real symmetric
D4a2.	Real nonsymmetric
D4a3.	Complex Hermitian
D4a4.	Complex non-Hermitian
D4a5.	Tridiagonal
D4a6.	Banded
D4a7.	Sparse
D4b.	Generalized eigenvalue problems (e.g., $Ax = \lambda Bx$)
D4b1.	Real symmetric
D4b2.	Real general
D4b3.	Complex Hermitian
D4b4.	Complex general
D4b5.	Banded
D4c.	Associated operations
D4c1.	Transform problem
D4c1a.	Balance matrix
D4c1b.	Reduce to compact form
D4c1b1.	Tridiagonal
D4c1b2.	Hessenberg
D4c1b3.	Other
D4c1c.	Standardize problem
D4c2.	Compute eigenvalues of matrix in compact form
D4c2a.	Tridiagonal
D4c2b.	Hessenberg
D4c2c.	Other
D4c3.	Form eigenvectors from eigenvalues
D4c4.	Back transform eigenvectors
D4c5.	Determine Jordan normal form
D5.	QR decomposition, Gram-Schmidt orthogonalization
D6.	Singular value decomposition
D7.	Update matrix decompositions
D7a.	LU
D7b.	Cholesky
D7c.	QR
D7d.	Singular value
D8.	Other matrix equations (e.g., $AX + XB = C$)
D9.	Singular, overdetermined or underdetermined systems of linear equations, generalized inverses

D9a. Unconstrained
D9a1. Least squares (L_2) solution
D9a2. Chebyshev (L_∞) solution
D9a3. Least absolute value (L_1) solution
D9a4. Other
D9b. Constrained
D9b1. Least squares (L_2) solution
D9b2. Chebyshev (L_∞) solution
D9b3. Least absolute value (L_1)
D9b4. Other
D9c. Generalized inverses

E. Interpolation

E1. Univariate data (curve fitting)
E1a. Polynomial splines (piecewise polynomials)
E1b. Polynomials
E1c. Other functions (e.g., rational, trigonometric)
E2. Multivariate data (surface fitting)
E2a. Gridded
E2b. Scattered
E3. Service routines for interpolation
E3a. Evaluation of fitted functions, including quadrature
E3a1. Function evaluation
E3a2. Derivative evaluation
E3a3. Quadrature
E3b. Grid or knot generation
E3c. Manipulation of basis functions (e.g., evaluation, change of basis)
E3d. Other

F. Solution of nonlinear equations

F1. Single equation
F1a. Polynomial
F1a1. Real coefficients
F1a2. Complex coefficients
F1b. Nonpolynomial
F2. System of equations
F3. Service routines (e.g., check user-supplied derivatives)

G. Optimization *(search also classes K, L8)*

G1. Unconstrained
G1a. Univariate
G1a1. Smooth function
G1a1a. User provides no derivatives
G1a1b. User provides first derivatives
G1a1c. User provides first and second derivatives
G1a2. General function (no smoothness assumed)
G1b. Multivariate
G1b1. Smooth function
G1b1a. User provides no derivatives
G1b1b. User provides first derivatives

G1b1c.	User provides first and second derivatives
G1b2.	General function (no smoothness assumed)
G2.	Constrained
G2a.	Linear programming
G2a1.	Dense matrix of constraints
G2a2.	Sparse matrix of constraints
G2b.	Transportation and assignments problem
G2c.	Integer programming
G2c1.	Zero/one
G2c2.	Covering and packing problems
G2c3.	Knapsack problems
G2c4.	Matching problems
G2c5.	Routing, scheduling, location problems
G2c6.	Pure integer programming
G2c7.	Mixed integer programming
G2d.	Network (for network reliability search class M)
G2d1.	Shortest path
G2d2.	Minimum spanning tree
G2d3.	Maximum flow
G2d3a.	Generalized networks
G2d3b.	Networks with side constraints
G2d4.	Test problem generation
G2e.	Quadratic programming
G2e1.	Positive definite Hessian (i.e., convex problem)
G2e2.	Indefinite Hessian
G2f.	Geometric programming
G2g.	Dynamic programming
G2h.	General nonlinear programming
G2h1.	Simple bounds
G2h1a.	Smooth function
G2h1a1.	User provides no derivatives
G2h1a2.	User provides first derivatives
G2h1a3.	User provides first and second derivatives
G2h1b.	General function (no smoothness assumed)
G2h2.	Linear equality or inequality constraints
G2h2a.	Smooth function
G2h2a1.	User provides no derivatives
G2h2a2.	User provides first derivatives
G2h2a3.	User provides first and second derivatives
G2h2b.	General function (no smoothness assumed)
G2h3.	Nonlinear constraints
G2h3a.	Equality constraints only
G2h3a1.	Smooth function and constraints
G2h3a1a.	User provides no derivatives
G2h3a1b.	User provides first derivatives of function and constraints
G2h3a1c.	User provides first and second derivatives of function and constraints
G2h3a2.	General function and constraints (no smoothness assumed)
G2h3b.	Equality and inequality constraints
G2h3b1.	Smooth function and constraints
G2h3b1a.	User provides no derivatives
G2h3b1b.	User provides first derivatives of function and constraints
G2h3b1c.	User provides first and second derivatives of function and constraints

G2h3b2. General function and constraints (no smoothness assumed)
G2i. Global solution to nonconvex problems
G3. Optimal control
G4. Service routines
G4a. Problem input (e.g., matrix generation)
G4b. Problem scaling
G4c. Check user-supplied derivatives
G4d. Find feasible point
G4e. Check for redundancy
G4f. Other

H. Differentiation, integration

H1. Numerical differentiation
H2. Quadrature (numerical evaluation of definite integrals)
H2a. One-dimensional integrals
H2a1. Finite interval (general integrand)
H2a1a. Integrand available via user-defined procedure
H2a1a1. Automatic (user need only specify required accuracy)
H2a1a2. Nonautomatic
H2a1b. Integrand available only on grid
H2a1b1. Automatic (user need only specify required accuracy)
H2a1b2. Nonautomatic
H2a2. Finite interval (specific or special type integrand including weight functions, oscillating and singular integrands, principal value integrals, splines, etc.)
H2a2a. Integrand available via user-defined procedure
H2a2a1. Automatic (user need only specify required accuracy)
H2a2a2. Nonautomatic
H2a2b. Integrand available only on grid
H2a2b1. Automatic (user need only specify required accuracy)
H2a2b2. Nonautomatic
H2a3. Semi-infinite interval (including $\exp -x$ weight function)
H2a3a. Integrand available via user-defined procedure
H2a3a1. Automatic (user need only specify required accuracy)
H2a3a2. Nonautomatic
H2a4. Infinite interval (including $\exp -x^2$ weight function)
H2a4a. Integrand available via user-defined procedure
H2a4a1. Automatic (user need only specify required accuracy)
H2a4a2. Nonautomatic
H2b. Multidimensional integrals
H2b1. One or more hyper-rectangular regions (includes iterated integrals)
H2b1a. Integrand available via user-defined procedure
H2b1a1. Automatic (user need only specify required accuracy)
H2b1a2. Nonautomatic
H2b1b. Integrand available only on grid
H2b1b1. Automatic (user need only specify required accuracy)
H2b1b2. Nonautomatic
H2b2. n-dimensional quadrature on a nonrectangular region
H2b2a. Integrand available via user-defined procedure
H2b2a1. Automatic (user need only specify required accuracy)
H2b2a2. Nonautomatic
H2b2b. Integrand available only on grid
H2b2b1. Automatic (user need only specify required accuracy)

H2b2b2.	Nonautomatic
H2c.	Service routines (e.g., compute weights and nodes for quadrature formulas)

I. Differential and integral equations

I1.	Ordinary differential equations (ODE's)
I1a.	Initial value problems
I1a1.	General, nonstiff or mildly stiff
I1a1a.	One-step methods (e.g., Runge-Kutta)
I1a1b.	Multistep methods (e.g., Adams predictor-corrector)
I1a1c.	Extrapolation methods (e.g., Bulirsch-Stoer)
I1a2.	Stiff and mixed algebraic- differential equations
I1b.	Multipoint boundary value problems
I1b1.	Linear
I1b2.	Nonlinear
I1b3.	Eigenvalue (e.g., Sturm-Liouville)
I1c.	Service routines (e.g., interpolation of solutions, error handling, test programs)
I2.	Partial differential equations
I2a.	Initial boundary value problems
I2a1.	Parabolic
I2a1a.	One spatial dimension
I2a1b.	Two or more spatial dimensions
I2a2.	Hyperbolic
I2b.	Elliptic boundary value problems
I2b1.	Linear
I2b1a.	Second order
I2b1a1.	Poisson (Laplace) or Helmholtz equation
I2b1a1a.	Rectangular domain (or topologically rectangular in the coordinate system)
I2b1a1b.	Nonrectangular domain
I2b1a2.	Other separable problems
I2b1a3.	Nonseparable problems
I2b1c.	Higher order equations (e.g., biharmonic)
I2b2.	Nonlinear
I2b3.	Eigenvalue
I2b4.	Service routines
I2b4a.	Domain triangulation *(search also class P)*
I2b4b.	Solution of discretized elliptic equations
I3.	Integral equations

J. Integral transforms

J1.	Trigonometric transforms including fast Fourier transforms
J1a.	One-dimensional
J1a1.	Real
J1a2.	Complex
J1a3.	Sine and cosine transforms
J1b.	Multidimensional
J2.	Convolutions
J3.	Laplace transforms
J4.	Hilbert transforms

K. Approximation *(search also class L8)*

K1. Least squares (L_2) approximation
K1a. Linear least squares *(search also classes D5, D6, D9)*
K1a1. Unconstrained
K1a1a. Univariate data (curve fitting)
K1a1a1. Polynomial splines (piecewise polynomials)
K1a1a2. Polynomials
K1a1a3. Other functions (e.g., trigonometric, user-specified)
K1a1b. Multivariate data (surface fitting)
K1a2. Constrained
K1a2a. Linear constraints
K1a2b. Nonlinear constraints
K1b. Nonlinear least squares
K1b1. Unconstrained
K1b1a. Smooth functions
K1b1a1. User provides no derivatives
K1b1a2. User provides first derivatives
K1b1a3. User provides first and second derivatives
K1b1b. General functions
K1b2. Constrained
K1b2a. Linear constraints
K1b2b. Nonlinear constraints
K2. Minimax (L_∞) approximation
K3. Least absolute value (L_1) approximation
K4. Other analytic approximations (e.g., Taylor polynomial, Pade)
K5. Smoothing
K6. Service routines for approximation
K6a. Evaluation of fitted functions, including quadrature
K6a1. Function evaluation
K6a2. Derivative evaluation
K6a3. Quadrature
K6b. Grid or knot generation
K6c. Manipulation of basis functions (e.g., evaluation, change of basis)
K6d. Other

L. Statistics, probability

L1. Data summarization
L1a. One-dimensional data
L1a1. Raw data
L1a1a. Location
L1a1b. Dispersion
L1a1c. Shape
L1a1d. Frequency, cumulative frequency
L1a1e. Ties
L1a3. Grouped data
L1b. Two dimensional data *(search also class L1c)*
L1c. Multi-dimensional data
L1c1. Raw data
L1c1b. Covariance, correlation
L1c1d. Frequency, cumulative frequency
L1c2. Raw data containing missing values *(search also class L1c1)*

L2.	Data manipulation
L2a.	Transform *(search also classes L10a1, N6, and N8)*
L2b.	Tally
L2c.	Subset
L2d.	Merge *(search also class N7)*
L2e.	Construct new variables (e.g., indicator variables)
L3.	Elementary statistical graphics *(search also class Q)*
L3a.	One-dimensional data
L3a1.	Histograms
L3a2.	Frequency, cumulative frequency, percentile plots
L3a3.	EDA (e.g., box-plots)
L3a4.	Bar charts
L3a5.	Pie charts
L3a6.	X_i vs. i (including symbol plots)
L3a7.	Lag plots (e.g., plots of X_i vs. X_{i-1})
L3b.	Two-dimensional data *(search also class L3e)*
L3b1.	Histograms (superimposed and bivariate)
L3b2.	Frequency, cumulative frequency
L3b3.	Scatter diagrams
L3b3a.	Y vs. X
L3b3b.	Symbol plots
L3b3c.	Lag plots (i.e., plots of X_i vs. Y_{i-j})
L3b4.	EDA
L3c.	Three-dimensional data *(search also class L3e)*
L3e.	Multi-dimensional data
L3e1.	Histograms
L3e2.	Frequency, cumulative frequency, percentile plots
L3e3.	Scatter diagrams
L3e3a.	Superimposed Y vs. X
L3e3c.	Superimposed X_i vs. i
L3e3d.	Matrices of bivariate scatter diagrams
L3e4.	EDA
L4.	Elementary data analysis
L4a.	One-dimensional data
L4a1.	Raw data
L4a1a.	Parametric analysis
L4a1a1.	Plots of empirical and theoretical density and distribution functions
L4a1a2.	Probability plots
L4a1a2b.	Beta, binomial
L4a1a2c.	Cauchy, chi-squared
L4a1a2d.	Double exponential
L4a1a2e.	Exponential, extreme value
L4a1a2f.	F distribution
L4a1a2g.	Gamma, geometric
L4a1a2h.	Halfnormal
L4a1a2l.	Lambda, logistic, lognormal
L4a1a2n.	Negative binomial, normal
L4a1a2p.	Pareto, Poisson
L4a1a2s.	Semicircular
L4a1a2t.	t distribution, triangular
L4a1a2u.	Uniform
L4a1a2w.	Weibull

L4a1a3. Probability plot correlation coefficient plots
L4a1a3c. Chi-squared
L4a1a3e. Extreme value
L4a1a3g. Gamma, geometric
L4a1a3l. Lambda
L4a1a3n. Normal
L4a1a3p. Pareto, Poisson
L4a1a3t. t distribution
L4a1a3w. Weibull
L4a1a4. Parameter estimates and tests
L4a1a4b. Binomial
L4a1a4e. Extreme value
L4a1a4n. Normal
L4a1a4p. Poisson
L4a1a4u. Uniform
L4a1a4w. Weibull
L4a1a5. Transformation selection (e.g., for normality)
L4a1a6. Tail and outlier analysis
L4a1a7. Tolerance limits
L4a1b. Nonparametric analysis
L4a1b1. Estimates and tests regarding location (e.g., median), dispersion, and shape
L4a1b2. Density function estimation
L4a1c. Goodness-of-fit tests
L4a1d. Analysis of a sequence of numbers *(search also class L10a)*
L4a3. Grouped and/or censored data
L4a4. Data sampled from a finite population
L4a5. Categorical data
L4b. Two dimensional data *(search also class L4c)*
L4b1. Pairwise independent data
L4b1a. Parametric analysis
L4b1a1. Plots of empirical and theoretical density and distribution functions
L4b1a4. Parameter estimates and hypothesis tests
L4b1b. Nonparametric analysis (e.g., rank tests)
L4b1c. Goodness-of-fit tests
L4b3. Pairwise dependent data
L4b4. Pairwise dependent grouped data
L4b5. Data sampled from a finite population
L4c. Multi-dimensional data *(search also classes L4b and L7a1)*
L4c1. Independent data
L4c1a. Parametric analysis
L4c1b. Nonparametric analysis
L4e. Multiple multi-dimensional data sets
L5. Function evaluation *(search also class C)*
L5a. Univariate
L5a1. Cumulative distribution functions, probability density functions
L5a1b. Beta, binomial
L5a1c. Cauchy, chi-squared
L5a1d. Double exponential
L5a1e. Error function, exponential, extreme value
L5a1f. F distribution
L5a1g. Gamma, general, geometric
L5a1h. Halfnormal, hypergeometric

L5a1k.	Kendall F statistic, Kolmogorov-Smirnov
L5a1l.	Lambda, logistic, lognormal
L5a1n.	Negative binomial, normal
L5a1p.	Pareto, Poisson
L5a1t.	t distribution
L5a1u.	Uniform
L5a1v.	Von Mises
L5a1w.	Weibull
L5a2.	Inverse distribution functions, sparsity functions
L5a2b.	Beta, binomial
L5a2c.	Cauchy, chi-squared
L5a2d.	Double exponential
L5a2e.	Error function, exponential, extreme value
L5a2f.	F distribution
L5a2g.	Gamma, general, geometric
L5a2h.	Halfnormal
L5a2l.	Lambda, logistic, lognormal
L5a2n.	Negative binomial, normal, normal order statistics
L5a2p.	Pareto, Poisson
L5a2t.	t distribution
L5a2u.	Uniform
L5a2w.	Weibull
L5b.	Multivariate
L5b1.	Cumulative multivariate distribution functions, probability density functions
L5b1n.	Normal
L5b2.	Inverse cumulative distribution functions
L5b2n.	Normal
L6.	Random number generation
L6a.	Univariate
L6a2.	Beta, binomial, Boolean
L6a3.	Cauchy, chi-squared
L6a4.	Double exponential
L6a5.	Exponential, extreme value
L6a6.	F distribution
L6a7.	Gamma, general (continuous, discrete), geometric
L6a8.	Halfnormal, hypergeometric
L6a12.	Lambda, logistic, lognormal
L6a14.	Negative binomial, normal, normal order statistics
L6a16.	Pareto, Pascal, permutations, Poisson
L6a19.	Samples, stable distribution
L6a20.	t distribution, time series, triangular
L6a21.	Uniform (continuous, discrete), uniform order statistics
L6a22.	Von Mises
L6a23.	Weibull
L6b.	Multivariate
L6b3.	Contingency table, correlation matrix
L6b5.	Experimental designs
L6b12.	Linear L_1 (least absolute value) approximation
L6b13.	Multinomial
L6b14.	Normal
L6b15.	Orthogonal matrix
L6b21.	Uniform

L6c. Service routines (e.g., seed)
L7. Analysis of variance (including analysis of covariance)
L7a. One-way
L7a1. Parametric
L7a2. Nonparametric
L7b. Two-way *(search also class L7d)*
L7c. Three-way (e.g., Latin squares) *(search also class L7d)*
L7d. Multi-way
L7d1. Balanced complete data (e.g., factorial designs)
L7d2. Balanced incomplete data
L7d3. General linear models (unbalanced data)
L7e. Multivariate
L7f. Generate experimental designs
L7g. Service routines
L8. Regression *(search also classes D5, D6, D9, G, K)*
L8a. Simple linear (i.e., $y = b_0 + b_1 x$) *(search also class L8h)*
L8a1. Ordinary least squares
L8a1a. Parameter estimation
L8a1a1. Unweighted data
L8a1a2. Weighted data
L8a1d. Inference (e.g., calibration) *(search also class L8a1a)*
L8a2. L_p for p different from 2 (e.g., least absolute value, minimax)
L8a3. Robust
L8a4. Errors in variables
L8b. Polynomial (e.g., $y = b_0 + b_1 x + b_2 x^2$) *(search also class L8c)*
L8b1. Ordinary least squares
L8b1a. Degree determination
L8b1b. Parameter estimation
L8b1b1. Not using orthogonal polynomials
L8b1b2. Using orthogonal polynomials
L8b1c. Analysis *(search also class L8b1b)*
L8b1d. Inference *(search also class L8b1b)*
L8c. Multiple linear (i.e., $y = b_0 + b_1 x_1 + \ldots + b_p x_p$)
L8c1. Ordinary least squares
L8c1a. Variable selection
L8c1a1. Using raw data
L8c1a2. Using correlation or covariance data
L8c1a3. Using other data
L8c1b. Parameter estimation *(search also class L8c1a)*
L8c1b1. Using raw data
L8c1b2. Using correlation data
L8c1c. Analysis *(search also classes L8c1a and L8c1b)*
L8c1d. Inference *(search also classes L8c1a and L8c1b)*
L8c2. Several regressions
L8c3. L_p for p different from 2
L8c4. Robust
L8c5. Measurement error models
L8c6. Models based on ranks
L8d. Polynomial in several variables
L8e. Nonlinear (i.e., $y = F(X, b)$) *(search also class L8h)*
L8e1. Ordinary least squares
L8e1a. Variable selection

L8e1b. Parameter estimation *(search also class L8e1a)*
L8e1b1. Unweighted data, user provides no derivatives
L8e1b2. Unweighted data, user provides derivatives
L8e1b3. Weighted data, user provides no derivatives
L8e1b4. Weighted data, user provides derivatives
L8e2. Ridge
L8e5. Measurement error models
L8f. Simultaneous (i.e., $Y = Xb$)
L8g. Spline (i.e., piecewise polynomial)
L8h. EDA (e.g., smoothing)
L8i. Service routines (e.g., matrix manipulation for variable selection)
L9. Categorical data analysis
L9a. 2-by-2 tables
L9b. Two-way tables *(search also class L9d)*
L9c. Log-linear model
L9d. EDA (e.g., median polish)
L10. Time series analysis *(search also class J)*
L10a. Univariate *(search also classes L3a6 and L3a7)*
L10a1. Transformations
L10a1a. Elementary *(search also class L2a)*
L10a1b. Stationarity *(search also class L8a1)*
L10a1c. Filters *(search also class K5)*
L10a1c1. Difference
L10a1c2. Symmetric linear (e.g., moving averages)
L10a1c3. Autoregressive linear
L10a1c4. Other
L10a1d. Taper
L10a2. Time domain analysis
L10a2a. Summary statistics
L10a2a1. Autocorrelations and autocovariances
L10a2a2. Partial autocorrelations
L10a2b. Stationarity analysis *(search also class L10a2a)*
L10a2c. Autoregressive models
L10a2c1. Model identification
L10a2c2. Parameter estimation
L10a2d. ARMA and ARIMA models (including Box-Jenkins methods)
L10a2d1. Model identification
L10a2d2. Parameter estimation
L10a2d3. Forecasting
L10a2e. State-space analysis (e.g., Kalman filtering)
L10a2f. Analysis of a locally stationary series
L10a3. Frequency domain analysis *(search also class J1)*
L10a3a. Spectral analysis
L10a3a1. Pilot analysis
L10a3a2. Periodogram analysis
L10a3a3. Spectrum estimation using the periodogram
L10a3a4. Spectrum estimation using the Fourier transform of the autocorrelation function
L10a3a5. Spectrum estimation using autoregressive models
L10a3a6. Spectral windows
L10a3b. Complex demodulation
L10b. Two time series *(search also classes L3b3c, L10c, and L10d)*
L10b2. Time domain analysis

L10b2a. Summary statistics (e.g., cross-correlations)
L10b2b. Transfer function models
L10b3. Frequency domain analysis *(search also class J1)*
L10b3a. Cross-spectral analysis
L10b3a2. Cross-periodogram analysis
L10b3a3. Cross-spectrum estimation using the cross-periodogram
L10b3a4. Cross-spectrum estimation using the Fourier transform of the cross-correlation or cross-covariance function
L10b3a6. Spectral functions
L10c. Multivariate time series *(search also classes J1, L3e3 and L10b)*
L10d. Two multi-channel time series
L11. Correlation analysis *(search also classes L4 and L13c)*
L12. Discriminant analysis
L13. Covariance structure models
L13a. Factor analysis
L13b. Principal components analysis
L13c. Canonical correlation
L14. Cluster analysis
L14a. One-way
L14a1. Unconstrained
L14a1a. Nested
L14a1a1. Joining (e.g., single link)
L14a1a2. Divisive
L14a1a3. Switching
L14a1a4. Predict missing values
L14a1b. Non-nested (e.g., K means)
L14a2. Constrained
L14b. Two-way
L14c. Display
L14d. Service routines (e.g., compute distance matrix)
L15. Life testing, survival analysis
L16. Multidimensional scaling
L17. Statistical data sets

M. Simulation, stochastic modeling *(search also classes L6 and L10)*

M1. Simulation
M1a. Discrete
M1b. Continuous (Markov models)
M2. Queueing
M3. Reliability
M3a. Quality control
M3b. Electrical network
M4. Project optimization (e.g., PERT)

N. Data handling *(search also class L2)*

N1. Input, output
N2. Bit manipulation
N3. Character manipulation
N4. Storage management (e.g., stacks, heaps, trees)
N5. Searching
N5a. Extreme value

N5b. Insertion position
N5c. On a key
N6. Sorting
N6a. Internal
N6a1. Passive (i.e. construct pointer array, rank)
N6a1a. Integer
N6a1b. Real
N6a1c. Character
N6a2. Active
N6a2a. Integer
N6a2b. Real
N6a2c. Character
N6b. External
N7. Merging
N8. Permuting

O. Symbolic computation

P. Computational geometry *(search also classes G and Q)*

Q. Graphics *(search also class L3)*

R. Service routines

R1. Machine-dependent constants
R2. Error checking (e.g., check monotonicity)
R3. Error handling
R3a. Set criteria for fatal errors
R3b. Set unit number for error messages
R3c. Other utilities
R4. Documentation retrieval

S. Software development tools

S1. Program transformation tools
S2. Static program analysis tools
S3. Dynamic program analysis tools

Z. Other

Appendix B
Changes from Version 1.2

Here we summarize how the GAMS Classification System has changed from version 1.2 which was published in [3]. A number of additional cosmetic changes were made to the text of the system; these are not enumerated here.

- *Subtree A3 (Real arithmetic)*
  A3a changed from Real to Standard precision. A3b (Double precision) removed.
- *Subtree A4 (Complex arithmetic)*
  A4a changed from Real to Standard precision. A4b (Double precision) removed.
- *D1a11 (Other vector operations)*
  New subclass.
- *D9 (Singular, overdetermined or underdetermined systems of linear equations, generalized inverses)*
  Tree refined, 11 new subclasses added.
- *E3 (Service routines for interpolation)*
  Tree refined, 14 new subclasses added.
- *Subtree F1 (Single nonlinear equations)*
  Revised to remove distinction between smooth and nonsmooth functions.
- *Subtree F2 (System of nonlinear equations)*
  All subclasses deleted, removing the distinction between smooth and nonsmooth functions.
- *K6 (Service routines for approximation)*
  Tree refined, 14 new subclasses added.
- *Subtree L (Statistics and Probability)*
  Substantially revised. Classes L1, L3, L4, L7, and L14 were revised to standardize the first level of subclasses (e.g., L1a, L1b, ...) as the dimension of the data; class L8 was revised so that the first level of its subclasses were functional form. Classes L1, L2, L3, L4, L7, L8, and L10 were revised to better reflect available software. Probability plots were moved from L3 to L4. Class L13 was completely revised. Classes L16 and L17 were added.
- *Subtree N6a1b (Internal sorting of real data)*
  All subclasses deleted, removing the distinction between single and double precision data.
- *Subtree N6a2b (Internal sorting of real data)*
  All subclasses deleted, removing the distinction between single and double precision data.
- *Subtree P (Computational Geometry)*
  All subclasses deleted.
- *Subtree Q (Graphics)*
  Subclass Q1 (Line printer graphics) deleted. Q has no subclasses in the revised system.

In addition, the text of the following classes was revised in order to clarify their purposes: C1, H2b1, H2b2, J1, J1a3.

Classes A3, A4, N6a1b and N6a2b all were changed to remove classes which referred to double precision to insure that both single and double precision versions of programs could always have the same classification.

Classes D9, E3, and K6 were refined because of a wealth of software now available for these problems. The subtrees E3 and K6 are exactly parallel; they provide homes for low-level routines for manipulating spline basis functions.

The subtrees P and Q were both trimmed so that they would be unrefined. In the case of P we felt that not enough software was currently available to adequately define the subject area. In the case of Q we did not feel that we possessed the expertise to adequately refine it.

NIST-114A
(REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4475

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
DECEMBER 1990

4. TITLE AND SUBTITLE
The Guide to Available Mathematical Software Problem Classification System

5. AUTHOR(S)
Ronald F. Boisvert, Sally E. Howe, and David K. Kahaner

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

A vast collection of reusable mathematical and statistical software parts is now available for use by scientists and engineers in their modeling efforts. This software represents a significant source of mathematical expertise, created and maintained at considerable expense. Unfortunately, the collection is so heterogeneous that it is a tedious and error-prone task simply to determine what software is available to solve a given problem. In mathematical problem-solving environments of the future such questions will be fielded by expert software advisory systems. One way for such systems to systematically associate available software with the problems they solve is to use a problem classification system. In this paper we describe a detailed tree-structured problem-oriented classification system appropriate for such use.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
classification system; software documentation; mathematical software; software advisory systems; software catalogs; statistical software

13. AVAILABILITY

- [X] UNLIMITED
- [ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
- [ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
- [X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
32

15. PRICE
A03

ELECTRONIC FORM